Notes on HyperKZG

In Gemini-PCS [BCHO22], a coefficient-form MLE polynomial corresponds to a univariate polynomial,

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = f_0 + f_1 X_0 + f_2 X_1 + f_3 X_0 X_1 + \cdots + f_{2^n-1} X_0 X_1 \cdots X_{n-1} \tag{1}$$

corresponds to a univariate polynomial:

$$f(X) = f_0 + f_1 X + f_2 X^2 + \cdots + f_{2^n-1} X^{2^n-1} \tag{2}$$

When we determine a public evaluation point $\vec{u} = (u_0, u_1, \ldots, u_{n-1})$, the value of the MLE polynomial $\tilde{f}$ at $\vec{u}$ can be expressed as the following Tensor Product:

$$\tilde{f}(u_0, u_1, \ldots, u_{n-1}) = \langle \vec{f}, \otimes_{i=0}^{n-1}(1, u_i) \rangle \tag{3}$$

Next, Gemini-PCS uses the Split-and-fold approach to convert the above equation into the correctness of evaluations of multiple univariate polynomials, which can be proven using KZG10.

However, MLE polynomials are usually represented in point-value form by default,

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = \sum_{i=0}^{2^n-1} a_i \cdot \tilde{eq}(\mathsf{bits}(i), (X_0, X_1, \ldots, X_{n-1})) \tag{4}$$

To use Gemini-PCS, the Prover needs to first convert the point-value form of MLE to the coefficient form mentioned above, i.e., calculate the $\vec{f}$ vector from the $\vec{a}$ vector. This conversion algorithm is similar to FFT computation, with a time complexity of $O(N \log N)$, where $N = 2^n$.

The idea of HyperKZG is to still utilize the core Split-and-fold approach of Gemini-PCS, but without the need for polynomial conversion similar to FFT. This may sound incredible at first, but the key point here is that MLE polynomials are essentially linear polynomials in multidimensional space. Whether in Evaluation-form or Coefficient-form, their computation process is actually a linear operation. At the same time, the Split-and-fold approach adopted by Gemini-PCS is also a mapping process that continuously reduces the dimensionality of high-dimensional space. Therefore, this Split-and-fold process can be transplanted to the Evaluation-form of MLE, achieving the same folding effect, but perfectly avoiding the complex calculations of polynomial Basis conversion.

1. Review of Gemini-PCS Principles

Gemini [BCHO22] provides a method for mapping MLE polynomials to univariate polynomials. Below is the definition of an MLE:

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = f_0 + f_1 X_0 + f_2 X_1 + f_3 X_0 X_1 + \cdots + f_{2^n-1} X_0 X_1 \cdots X_{n-1} \tag{5}$$

If we use a coefficient vector $\vec{f}$ of length $N = 2^n$ to represent $\tilde{f}$, then we can define a Univariate polynomial $f(X)$ that has the same coefficients as $\tilde{f}(X_0, \ldots, X_{n-1})$:

$$f(X) = f_0 + f_1 X + f_2 X^2 + \cdots + f_{2^n-1} X^{2^n-1} \tag{6}$$

If we substitute $X = -X$ into the above equation, we get:

$$f(-X) = f_0 - f_1 X + f_2 X^2 - \cdots - f_{2^n-1} X^{2^n-1} \tag{7}$$

Then by adding and subtracting these two equations respectively, we can get:

$$f(X) + f(-X) = 2(f_0 + f_2 X^2 + \cdots + f_{2^n-2} X^{2^n-2}) \tag{8}$$

$$f(X) - f(-X) = 2X(f_1 + f_3 X^2 + \cdots + f_{2^n-1} X^{2^n-2}) \tag{9}$$

Now let's observe the "partial computation" Partial Evaluation of $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$, i.e., instantiating only one unknown $X_0 = u_0$:

$$\begin{aligned} \tilde{f}(u_0, X_1, \ldots, X_{n-1}) &= f_0 + f_1 u_0 + f_2 X_1 + f_3 u_0 X_1 + f_4 X_2 + f_5 u_0 X_2 + f_6 X_1 X_2 + f_7 u_0 X_1 X_2 + \cdots + f_{2^n-1} u_0 X_1 \cdots X_{n-1} \\ &= (f_0 + f_1 u_0) + (f_2 + f_3 u_0) X_1 + (f_4 + f_5 u_0) X_2 + (f_6 + f_7 u_0) X_1 X_2 + \cdots + (f_{2^n-2} + f_{2^n-1} u_0) X_1 \cdots X_{n-1} \\ &= \tilde{f}^{(1)}(X_1, X_2, \ldots, X_{n-1}) \end{aligned}$$

Here, the coefficient vector of $\tilde{f}^{(1)}(X_1, X_2, \ldots, X_{n-1})$ is

$$(f_0 + f_1 u_0), (f_2 + f_3 u_0), (f_4 + f_5 u_0), (f_6 + f_7 u_0), \ldots, (f_{2^n-2} + f_{2^n-1} u_0) \tag{11}$$

It is exactly the same as the coefficient vector of the following UniPoly:

$$f^{(1)}(X) = (f_0 + f_1 u_0) + (f_2 + f_3 u_0) X + (f_4 + f_5 u_0) X^2 + \cdots + (f_{2^n-2} + f_{2^n-1} u_0) X^{2^{n-1}-1} \tag{12}$$

And the univariate polynomial $f^{(1)}(X)$ plus $f(X)$, $f(-X)$, the three exactly satisfy the following relationship:

$$\begin{aligned} f^{(1)}(X^2) &= (f_0 + f_1 u_0) + (f_2 + f_3 u_0) X^2 + (f_4 + f_5 u_0) X^4 + \cdots + (f_{2^n-2} + f_{2^n-1} u_0) X^{2^n-1} \\ &= \frac{1}{2}(f(X) + f(-X)) + u_0 \cdot \frac{1}{2X}(f(X) - f(-X)) \end{aligned} \tag{13}$$

So, if we want to prove that $\tilde{f}^{(1)}(X_1, \ldots, X_{n-1})$ is the Partial Evaluation of $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$, we only need to prove that the above equation holds.

Similarly, if we want to prove $\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = v$, we can introduce several intermediate results, namely Partial Evaluated MLE polynomials, and their isomorphic mappings to univariate polynomials

$$\begin{aligned}
\tilde{f}^{(0)}(X_0, X_1, \ldots, X_{n-1}) &\mapsto f^{(0)}(X) \\
\tilde{f}^{(1)}(u_0, X_1, \ldots, X_{n-1}) &\mapsto f^{(1)}(X) \\
\tilde{f}^{(2)}(u_0, u_1, \ldots, X_{n-1}) &\mapsto f^{(2)}(X) \\
&\vdots \\
\tilde{f}^{(n-1)}(u_0, u_1, \ldots, u_{n-2}, X_{n-1}) &\mapsto f^{(n-1)}(X) \\
\tilde{f}^{(n)}(u_0, u_1, \ldots, u_{n-2}, u_{n-1}) &\mapsto f^{(n)}(X)
\end{aligned} \tag{14}$$

where the last $f^{(n)}(X)$ is a constant polynomial, which is exactly the complete computation result of $\tilde{f}(u_0, u_1, \ldots, u_{n-1})$, i.e., $f^{(n)}(X) = v$.

And, these introduced univariate polynomials $f^{(0)}(X), \ldots, f^{(n-1)}(X)$ satisfy the following relationship between every two adjacent items:

$$f^{(i+1)}(X^2) = \frac{f^{(i)}(X) + f^{(i)}(-X)}{2} + u_i \cdot \frac{f^{(i)}(X) - f^{(i)}(-X)}{2X} \tag{15}$$

where $\tilde{f}^{(n)}(u_0, u_1, \ldots, u_{n-2}) = v$ is the final MLE computation result, and $h^{(0)}(X)$ is the UniPoly isomorphic to $\tilde{f}$: $h^{(0)}(X) = f(X)$.

Back to our proof goal: $\tilde{f}^{(n)}(u_0, u_1, \ldots, u_{n-1}) = v$, we split the proof of this computation process into the following steps:

- Construct a Univariate polynomial $f(X)$ such that its coefficient vector is equal to the coefficients of $\tilde{f}$, and construct the polynomial commitment $\mathsf{cm}(f(X))$
- The polynomial $\tilde{f}$ computation process includes n steps of partial computation, each intermediate partial computation will produce a new MLE polynomial: $\tilde{f}^{(1)}, \tilde{f}^{(2)}, \ldots, \tilde{f}^{(n-1)}$
- Prove that the Univariate polynomials corresponding to these intermediate MLEs satisfy a recursive relationship, which is randomly sampled and checked through the random number β provided by the Verifier:

$$f^{(i+1)}(\beta^2) = \frac{f^{(i)}(\beta) + f^{(i)}(-\beta)}{2} + u_i \cdot \frac{f^{(i)}(\beta) - f^{(i)}(-\beta)}{2\beta} \tag{16}$$

- Prove that $f^{(n)}(\beta^2) = v$
- Prove that all Univariate polynomials $\{f^{(i)}(X)\}$ are correctly evaluated at $X = \beta, X = -\beta, X = \beta^2$

2. Linear Folding of Evaluation-form

If we use the evaluation-form of MLE in the PIOP proof system, then we need a conversion operation similar to FFT to convert it to the coefficient-form of MLE. The complexity of this conversion operation is $O(N \log N)$.

In Nova's implementation, Setty provided an improved scheme for HyperKZG. It utilizes a general technique behind the Gemini PCS scheme, which is independent of whether $\tilde{f}$ is in Evaluation-form or Coefficient-form, as long as they can split the calculation process into multiple steps of linear operations.

Reviewing the MLE operation evaluation proof introduced in the Gemini paper in the previous section, it decomposes the Evaluation process of $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ into $\log n$ steps, then maps the coefficient vector of each intermediate MLE to the coefficients of a UniPoly, and then proves the relationship between these UniPolys to ensure the correctness of $\tilde{f}$ operation.

For an MLE polynomial $\tilde{f}(\vec{X})$ in Evaluation-form, let's look at its evaluation process:

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = a_0 E_0(X_0, X_1, \ldots, X_{n-1}) + a_1 E_1(X_0, X_1, \ldots, X_{n-1}) + \cdots + a_{2^n-1} E_{2^n-1}(X_0, X_1, \ldots, X_{n-1}) \tag{17}$$

Here $E_i(\vec{X})$ are Lagrange Polynomials, defined as follows:

$$E_i(X_0, X_1, \ldots, X_{n-1}) = \prod_{j=0}^{n-1} \left(\mathsf{bits}(i)_j \cdot X_j + (1 - \mathsf{bits}(i)_j)(1 - X_j)\right) \tag{18}$$

where $\mathsf{bits}(i)_j$ is the j-th bit of the binary representation of i (note that Big-endian representation is used here). For example, if $i = 5$, its binary representation is 101, then $\mathsf{bits}(5)_0 = 1$, $\mathsf{bits}(5)_1 = 0$, $\mathsf{bits}(5)_2 = 1$.

It's easy to see from the definition that $E_i(\vec{X})$ satisfies the following splitting property (Tensor Structure):

$$E_{i\|j}(\vec{X}, \vec{Y}) = E_i(\vec{X}) \cdot E_i(\vec{Y}) \tag{19}$$

where $k = i \parallel j$ is the splitting of the binary bit vector of k. For example, $i = 23$, its binary representation is 10111, which can be split into $10 \parallel 111$, or written as $23 = 2 \parallel 7$. According to the splitting property, we can get:

$$E_{i\|b}(X_0, (X_1, \ldots, X_{n-1})) = E_b(X_0) \cdot E_i(X_1, \ldots, X_{n-1}) \tag{20}$$

Then let's observe what $\tilde{f}$ looks like after one Partial Evaluation, let $X_0 = u_0$, $\vec{X} = (X_1, \ldots, X_{n-1})$:

$$
\begin{aligned}
\tilde{f}(u_0, \vec{X}) &= a_0 E_0(u_0, \vec{X}) + a_1 E_1(u_0, \vec{X}) + \cdots + a_{2^n-2} E_{2^n-2}(u_0, \vec{X}) + a_{2^n-1} E_{2^n-1}(u_0, \vec{X}) \\
&= \Big(a_0 E_0(u_0)\Big) \cdot E_0(\vec{X}) + \Big(a_1 E_1(u_0)\Big) \cdot E_0(\vec{X}) + \cdots + \Big(a_{2^n-1} E_0(u_0)\Big) \cdot E_{2^{n-1}-1}(\vec{X}) + \Big(a_{2^n-1} E_1(u_0)\Big) \cdot E_{2^{n-1}-1}(\vec{X}) \\
&= \Big(a_0(1-u_0)\Big) \cdot E_0(\vec{X}) + \Big(a_1 u_0\Big) \cdot E_0(\vec{X}) + \cdots + \Big(a_{2^n-2}(1-u_0)\Big) \cdot E_{2^{n-1}-1}(\vec{X}) + \Big(a_{2^n-1} u_0\Big) \cdot E_{2^{n-1}-1}(\vec{X}) \\
&= \Big(a_0(1-u_0)\Big) \cdot E_0(\vec{X}) + \Big(a_1 u_0\Big) \cdot E_0(\vec{X}) + \cdots + \Big(a_{2^n-2}(1-u_0)\Big) \cdot E_{2^{n-1}-1}(\vec{X}) + \Big(a_{2^n-1} u_0\Big) \cdot E_{2^{n-1}-1}(\vec{X}) \\
&= \Big(a_0(1-u_0) + a_1 u_0\Big) \cdot E_0(\vec{X}) + \cdots + \Big(a_{2^n-2}(1-u_0) + a_{2^n-1} u_0\Big) \cdot E_{2^{n-1}-1}(\vec{X}) \\
&= \tilde{h}^{(1)}(\vec{X})
\end{aligned}
\tag{21}
$$

We can see that the Evaluation point value vector of $\tilde{h}^{(1)}(\vec{X})$ is:

$$
\Big(a_0(1-u_0) + a_1 u_0\Big), \Big(a_2(1-u_0) + a_3 u_0\Big), \ldots, \Big(a_{2^n-2}(1-u_0) + a_{2^n-1} u_0\Big) \tag{22}
$$

Comparing with the coefficient vector of $\tilde{f}(u_0, \vec{X})$, we will find that both are only half the original length, but the halving method is different. The former is $\big((1-u)\cdot a + u \cdot b\big)$, while the latter is $\big(a + u \cdot b\big)$. This new halving method does not prevent us from using Gemini-PCS technology to ensure the correctness of Split-and-fold, but perfectly avoids the complex calculations of polynomial Basis conversion. We can still introduce $n-1$ partially computed MLE polynomials, and map their Evaluations to the coefficients of multiple UniPolys:

$$
\begin{aligned}
\tilde{h}^{(0)}(X_0, X_1, \ldots, X_{n-1}) &\mapsto h^{(0)}(X) \\
\tilde{h}^{(1)}(u_0, X_1, \ldots, X_{n-1}) &\mapsto h^{(1)}(X) \\
\tilde{h}^{(2)}(u_0, u_1, \ldots, X_{n-1}) &\mapsto h^{(2)}(X) \\
&\vdots \\
\tilde{h}^{(n-1)}(u_0, u_1, \ldots, u_{n-1}) &\mapsto h^{(n-1)}(X)
\end{aligned}
\tag{23}
$$

where $\tilde{h}^{(0)}(\vec{X}) = \tilde{f}(\vec{X})$, and $\tilde{h}^{(n-1)}(u_0, u_1, \ldots, u_{n-1}) = v$,

And give a "similar" recursive relationship that their Evaluation forms satisfy:

$$
h^{(i+1)}(X^2) = (1-u_i) \cdot \frac{h^{(i)}(X) + h^{(i)}(-X)}{2} + u_i \cdot \frac{h^{(i)}(X) - h^{(i)}(-X)}{2X} \tag{24}
$$

Therefore, the Verifier can then send a unique random challenge point $X = \beta$ to check whether $\{h^{(i)}(\beta)\}$ satisfy the recursive relationship defined by the above equation. This is where the KZG10 PCS scheme for Univariate polynomials can be connected to complete the rest of the proof.

3. Protocol Description

This section provides a description of the protocol flow. The protocol is to prove that the computation result of an MLE polynomial $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ at a given point $(u_0, u_1, \ldots, u_{n-1})$ equals v.

Witness Input:

1. $\vec{a} = (a_0, a_1, \ldots, a_{2^n-1})$: The coefficient vector of polynomial $f(X)$.

$$
f(X) = a_0 + a_1 X + a_2 X^2 + \cdots + a_{2^n-1} X^{2^n-1} \tag{25}
$$

Public Input:

1. C_f: The commitment of the isomorphic polynomial $f(X)$ of MLE polynomial $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$,

$$
C_f = \mathsf{KZG10.Commit}(f(X)) \tag{26}
$$

2. $\vec{u} = (u_0, u_1, \ldots, u_{n-1})$: The coordinates of the evaluation point
3. $v = \tilde{f}(u_0, u_1, \ldots, u_{n-1})$: The result of the computation

Round 1

1. Prover computes $h^{(1)}(X), h^{(2)}(X), \ldots, h^{(n-1)}(X)$

$$
\begin{aligned}
h^{(1)}(X) &= ((1-u_0)a_0 + u_0 a_1) + ((1-u_0)a_2 + u_0 a_3)X + \cdots + ((1-u_0)a_{2^n-2} + u_0 a_{2^n-1}) X^{2^{n-1}-1} \\
h^{(2)}(X) &= ((1-u_1)a_0^{(1)} + u_1 a_1^{(1)}) + ((1-u_1)a_2^{(1)} + u_1 a_3^{(1)})X + \ldots + ((1-u_1)a_{2^{n-1}-2}^{(1)} + u_1 a_{2^{n-1}-1}^{(1)}) X^{2^{n-2}-1} \\
&\vdots \\
h^{(n-1)}(X) &= ((1-u_{n-1})a_0^{(n-2)} + u_{n-1} a_1^{(n-2)}) + ((1-u_{n-1})a_2^{(n-2)} + u_{n-1} a_3^{(n-2)})X
\end{aligned}
\tag{27}
$$

2. Prover outputs commitments $\big(C_{h^{(1)}}, C_{h^{(2)}}, \ldots, C_{h^{(n-1)}}\big)$

Round 2

1. Verifier sends random challenge number $\beta \in \mathbb{F}$,

2. Prover computes and sends $\left(h^{(0)}(\beta), h^{(0)}(-\beta)\right)$, $\left(h^{(1)}(\beta), h^{(1)}(-\beta), h^{(1)}(\beta^2)\right)$, $\left(h^{(2)}(\beta), h^{(2)}(-\beta), h^{(2)}(\beta^2)\right)$, …, $\left(h^{(n-2)}(\beta), h^{(n-2)}(-\beta), h^{(n-2)}(\beta^2)\right)$
3. Prover proves the correctness of the above Evaluations and sends proofs: $\left(\pi_{0,\beta}, \pi_{0,-\beta}\right)$, $\left(\pi_{1,\beta}, \pi_{1,-\beta}, \pi_{1,\beta^2}\right)$, …, $\left(\pi_{n-1,\beta}, \pi_{n-1,-\beta}, \pi_{n-1,\beta^2}\right)$

Verification

1. Verify if the values of $h^{(0)}, \ldots, h^{(n-1)}$ at $X = \beta$, $X = -\beta$ and $X = \beta^2$ satisfy the recursive formula:

$$\begin{aligned} h^{(1)}(\beta^2) &\overset{?}{=} \frac{h^{(0)}(\beta) + h^{(0)}(-\beta)}{2} + u_0 \cdot \frac{h^{(0)}(\beta) - h^{(0)}(-\beta)}{2\beta} \\ h^{(2)}(\beta^2) &\overset{?}{=} \frac{h^{(1)}(\beta) + h^{(1)}(-\beta)}{2} + u_1 \cdot \frac{h^{(1)}(\beta) - h^{(1)}(-\beta)}{2\beta} \\ &\vdots \\ h^{(n-1)}(\beta^2) &\overset{?}{=} \frac{h^{(n-2)}(\beta) + h^{(n-2)}(-\beta)}{2} + u_{n-2} \cdot \frac{h^{(n-2)}(\beta) - h^{(n-2)}(-\beta)}{2\beta} \\ v &\overset{?}{=} \frac{h^{(n-1)}(\beta) + h^{(n-1)}(-\beta)}{2} + u_{n-1} \cdot \frac{h^{(n-1)}(\beta) - h^{(n-1)}(-\beta)}{2\beta} \end{aligned} \tag{28}$$

2. Based on $C_f, C_{h^{(1)}}, C_{h^{(2)}}, \ldots, C_{h^{(n-1)}}$, verify the correctness of polynomial evaluation operations:

$$\begin{aligned} &\mathsf{KZG10.Verify} \quad (C_f, \quad \beta, \quad h^{(0)}(\beta), \quad \pi_{0,\beta}) \quad \overset{?}{=} 1 \\ &\mathsf{KZG10.Verify} \quad (C_f, \quad -\beta, \quad h^{(0)}(-\beta), \quad \pi_{0,-\beta}) \quad \overset{?}{=} 1 \\ &\mathsf{KZG10.Verify} \quad (C_{h^{(1)}}, \quad \beta, \quad h^{(1)}(\beta), \quad \pi_{1,\beta}) \quad \overset{?}{=} 1 \\ &\mathsf{KZG10.Verify} \quad (C_{h^{(1)}}, \quad -\beta, \quad h^{(1)}(-\beta), \quad \pi_{1,-\beta}) \quad \overset{?}{=} 1 \\ &\mathsf{KZG10.Verify} \quad (C_{h^{(1)}}, \quad \beta^2, \quad h^{(1)}(\beta^2), \quad \pi_{1,\beta^2}) \quad \overset{?}{=} 1 \\ &\qquad\qquad\qquad\qquad\qquad \vdots \\ &\mathsf{KZG10.Verify} \quad (C_{h^{(n-1)}}, \quad \beta, \quad h^{(n-1)}(\beta), \quad \pi_{n-1,\beta}) \quad \overset{?}{=} 1 \\ &\mathsf{KZG10.Verify} \quad (C_{h^{(n-1)}}, \quad -\beta, \quad h^{(n-1)}(-\beta), \quad \pi_{n-1,-\beta}) \quad \overset{?}{=} 1 \\ &\mathsf{KZG10.Verify} \quad (C_{h^{(n-1)}}, \quad \beta^2, \quad h^{(n-1)}(\beta^2), \quad \pi_{n-1,\beta^2}) \quad \overset{?}{=} 1 \end{aligned} \tag{29}$$

4. Protocol Optimization

There are several points that can be optimized in the above protocol:

1. The Prover doesn't need to send all computation values at $X = \beta^2$ except for $h^{(0)}(\beta^2)$, because the Verifier can calculate them through the following recursive relationship. This can reduce the Prover's communication volume, and at the same time, the Verifier's calculation process is equivalent to performing verification simultaneously, thus saving the recursive formula verification process.

$$h^{(i+1)}(\beta^2) = \frac{h^{(i)}(\beta) + h^{(i)}(-\beta)}{2} + u_i \cdot \frac{h^{(i)}(\beta) - h^{(i)}(-\beta)}{2\beta} \tag{30}$$

2. The Prover can aggregate $h^{(0)}(X), \ldots, h^{(n-1)}(X)$ together through a random number γ, obtaining $h(X)$, and then prove the values of $h(X)$ at $X = \beta, -\beta, \beta^2$. This can avoid the Prover sending $3n - 1$ independent KZG10 Evaluation proofs, and only need to send three Evaluation proofs.

Below is the protocol description after optimization

Round 1

Prover computes $h^{(1)}(X), h^{(2)}(X), \ldots, h^{(n-1)}(X)$

$$\begin{aligned} h^{(1)}(X) &= ((1-u_0)a_0 + u_0 a_1) + ((1-u_0)a_2 + u_0 a_3)X + \cdots + ((1-u_0)a_{2^n-2} + u_0 a_{2^n-1})X^{2^{n-1}-1} \\ h^{(2)}(X) &= ((1-u_1)a_0^{(1)} + u_1 a_1^{(1)}) + ((1-u_1)a_2^{(1)} + u_1 a_3^{(1)})X + \ldots + ((1-u_1)a_{2^{n-1}-2}^{(1)} + u_1 a_{2^{n-1}-1}^{(1)})X^{2^{n-2}-1} \\ &\vdots \\ h^{(n-1)}(X) &= ((1-u_{n-1})a_0^{(n-2)} + u_{n-1} a_1^{(n-2)}) + ((1-u_{n-1})a_2^{(n-2)} + u_{n-1} a_3^{(n-2)})X \end{aligned} \tag{31}$$

Prover sends polynomial commitments $\left(C_{h^{(1)}}, C_{h^{(2)}}, \ldots, C_{h^{(n-1)}}\right)$

Round 2

1. Verifier sends random challenge number $\beta \in \mathbb{F}$,

2. Prover computes and sends $\left(h^{(0)}(\beta), h^{(0)}(-\beta), h^{(0)(\beta^2)}\right)$, $\left(h^{(1)}(\beta), h^{(1)}(-\beta)\right)$, $\left(h^{(2)}(\beta), h^{(2)}(-\beta)\right)$, …, $\left(h^{(n-1)}(\beta), h^{(n-1)}(-\beta)\right)$

Round 3

1. Verifier sends random challenge number $\gamma \in \mathbb{F}$,
2. Prover computes aggregated polynomial $h(X)$,

$$h(X) = h^{(0)}(X) + \gamma \cdot h^{(1)}(X) + \cdots + \gamma^{n-1} \cdot h^{(n-1)}(X) \tag{32}$$

3. Prover computes $v_\beta, v_{-\beta}, v_{\beta^2}$

$$\begin{aligned} v_\beta &= h^{(0)}(\beta) + \gamma \cdot h^{(1)}(\beta) + \cdots + \gamma^{n-1} \cdot h^{(n-1)}(\beta) \\ v_{-\beta} &= h^{(0)}(-\beta) + \gamma \cdot h^{(1)}(-\beta) + \cdots + \gamma^{n-1} \cdot h^{(n-1)}(-\beta) \\ v_{\beta^2} &= h^{(0)}(\beta^2) + \gamma \cdot h^{(1)}(\beta^2) + \cdots + \gamma^{n-1} \cdot h^{(n-1)}(\beta^2) \end{aligned} \tag{33}$$

4. Prover computes $h^*(x)$

$$h^*(X) = h(\beta) \cdot L_\beta(X) + h(-\beta) \cdot L_{-\beta}(X) + h(\beta^2) \cdot L_{\beta^2}(X) \tag{34}$$

Here, assume Domain $D = \{\beta, -\beta, \beta^2\}$, and $\{L_\beta(X), L_{-\beta}(X), L_{\beta^2}(X)\}$ are Lagrange Polynomials on D. Then $h^*(X)$ satisfies the following equation: there exists a quotient polynomial $q(X)$ such that

$$\mathsf{EQ1}: h(X) - h^*(X) = q(X) \cdot (X - \beta)(X + \beta)(X - \beta^2) \tag{35}$$

5. Prover computes quotient polynomial $q(X)$ and sends its polynomial commitment $C_q = \mathsf{cm}(q(X))$

Round 4

1. Verifier sends random challenge number $\zeta \in \mathbb{F}_p$
2. Prover computes linearization polynomial $r_\zeta(X)$ of $\mathsf{EQ1}$, satisfying $r_\zeta(\zeta) = 0$,

$$r_\zeta(X) = h(X) - h^*(\zeta) - q(X) \cdot (\zeta - \beta)(\zeta + \beta)(\zeta - \beta^2) \tag{36}$$

3. Prover sends the commitment of the linearization polynomial $C_r = [r(x)]_1$
4. Prover computes quotient polynomial $w(X)$ satisfying:

$$w(X) = \frac{r_\zeta(X)}{X - \zeta} \tag{37}$$

Prover sends $C_w = [w(x)]_1$

Verification

1. Compute $\left(h^{(1)}(\beta^2), h^{(2)}(\beta^2), \ldots, h^{(n-1)}(\beta^2)\right)$,

$$\begin{aligned} h^{(1)}(\beta^2) &= \frac{h^{(0)}(\beta) + h^{(0)}(-\beta)}{2} + u_0 \cdot \frac{h^{(0)}(\beta) - h^{(0)}(-\beta)}{2\beta} \\ h^{(2)}(\beta^2) &= \frac{h^{(1)}(\beta) + h^{(1)}(-\beta)}{2} + u_1 \cdot \frac{h^{(1)}(\beta) - h^{(1)}(-\beta)}{2\beta} \\ &\vdots \\ h^{(n-1)}(\beta^2) &= \frac{h^{(n-2)}(\beta) + h^{(n-2)}(-\beta)}{2} + u_{n-2} \cdot \frac{h^{(n-2)}(\beta) - h^{(n-2)}(-\beta)}{2\beta} \end{aligned} \tag{38}$$

2. Compute $h(\beta), h(-\beta), h(\beta^2)$,

$$\begin{aligned} h(\beta) &= h^{(0)}(\beta) + \gamma \cdot h^{(1)}(\beta) + \cdots + \gamma^{n-1} \cdot h^{(n-1)}(\beta) \\ h(-\beta) &= h^{(0)}(-\beta) + \gamma \cdot h^{(1)}(-\beta) + \cdots + \gamma^{n-1} \cdot h^{(n-1)}(-\beta) \\ h(\beta^2) &= h^{(0)}(\beta^2) + \gamma \cdot h^{(1)}(\beta^2) + \cdots + \gamma^{n-1} \cdot h^{(n-1)}(\beta^2) \end{aligned} \tag{39}$$

3. Compute the value of $c(X)$ at $X = \zeta$, $c(\zeta)$,
4. Compute $C_h = C_f + \gamma \cdot C_{h^{(1)}} + \gamma^2 \cdot C_{h^{(2)}} + \cdots + \gamma^{n-1} \cdot C_{h^{(n-1)}}$
5. Compute the commitment of $C_r = [r_\zeta(x)]_1$:

$$C_r = [r_\zeta(x)]_1 = C_h - c(\zeta) \cdot [1]_1 - (\zeta - \beta)(\zeta + \beta)(\zeta - \beta^2) \cdot C_q \tag{40}$$

4. Verify the relationship between C_h and C_w:

$$e(C_r + \zeta \cdot C_w, [1]_2) \stackrel{?}{=} e(C_w, [x]_2) \tag{41}$$

5. Performance Analysis

Proof size: $(n+1) \cdot \mathbb{G}_1 + (2n+1) \cdot \mathbb{F}$

$$\pi = \left(C_{h^{(1)}}, C_{h^{(2)}}, \ldots, C_{h^{(n-1)}}, C_q, C_w, \{h^{(i)}(\beta), h^{(i)}(-\beta)\}_{i=0}^{n-1}, h^{(0)}(\beta^2)\right) \tag{42}$$

Verifier Cost: $(2n+2) \cdot \mathsf{EccMul}^{\mathbb{G}_1} + (3n) \cdot \mathbb{F} + 2 \cdot \mathsf{Pairing}$

1. $2n \cdot \mathbb{F}$. Mult
2. Compute $h(\beta), h(-\beta), h(\beta^2)$: $3n \cdot \mathbb{F}$. Mult
3. Compute $c(\zeta)$: $O(1) \cdot \mathbb{F}$. Mult
4. Compute C_h: $n \cdot \mathbb{G}_1$ Scalar Multiplication
5. Compute P: $2 \cdot \mathbb{G}_1$ Scalar Multiplication

References